Service Abstraction

more...

Objects

The dynamic neighborhood model is based on an extensible list of object types (classes in OOP-notation, but we are not going to paint our model object-oriented. The ideas presented here are independent of the programming model). The current implementation covers 3 types of objects:

- locations
- persons, and
- communication.

Each instance - including locations - has a visibility-function. The function represents the strength of an object's presence in the space created by URLs.

Object Identification

Objects are identified by type and name. This name is globally unique for each object type. The name can be tailored to the object type for better readability and to simplify addressing of external resources. Locations are the most important resource external to the dynamic neighborhood model. The concept of URLs and URNs is imported from the Web. URLs are used as names of Location objects.

Person objects and communication objects are created by the components of the dynamic neighborhood service. Their names are constructed unique by including the host name of the dynamic neighborhood server, which creates the object. An example for a person's identifier is p123@servicehost.domain:serviceport. Implementations may use other globally unique ASCII-strings, e.g. email addresses of users, if they are known to the service.

Variables

Each object has instance variables to store individual information about the object. Part of the information such as the name is entered by users. Some information is extracted by various means and the third part is computed by the service at run-time. The service specification defines a data type for each of the variables.

The most important variables of the persons are: icon, nickname, locations, and neighbors. The variables "icon" and "nickname" are used to represent persons at the user interface level. Other variables control the computation of the visible area and the visibility-function. The "neighbors"-variable is computed by the service frequently. It contains the all-important list of visible persons in the neighborhood.

Location objects have the variables icon, links and persons. The icon is used to represent the page in an advanced user interface as a thumbnail image. The "links"-variable contains a list of hypertext references of the document. It also contains properties of the links, e.g. distance. The "persons"-variable is the counterpart to the "locations"-variable of persons. This does not necessarily mean that the information is stored twice.

Methods

Objects and variables are accessed through a set of methods. Available methods are:

- GET: retrieve the content of a variable,
- PUT: assign data to a variable,
- ADD and DELETE: add and remove data partially,
- SUBSCRIBE: subscribe for notifications of changes of a variable's content,
- UNSUBSCRIBE: end a subscription,
- ASSOCIATE and DISSOCIATE: see below.

Information between components of the neighborhood service is requested and exchanged via commands. Most of the commands are data-related. They manipulate or evaluate instance variables. Variable manipulation commands consist of an object, the object's name, the variable name, the method, and optional data.

Associations

The ASSOCIATE and DISSOCIATE methods work on objects. They are used to put objects into relation. The proposed neighborhood model uses directed 2-fold associations. Each association consists of 2 objects, the source- and the target-object.

ASSOCIATE-commands are directed to the target-object. They are comprised of the target-object identification, the source-object identification and attributes containing details of the association. The objects can be of any type.

For better readability we introduced aliases for associations between specific objects:

- Location-Location: LINK and UNLINK mark hypertext references between documents. This is similar to commands used by distributed hyperlink maintenance systems like [PitJon96].
- Person-Location: ENTER and LEAVE. These commands are generated on entry and exit of Web locations. The exact times are to be determined by information gathering components. Persons can be registered with more than one location.
- Person-Communication: JOIN and LEAVE let persons enter (and leave) communication channels.
- Person-Person: SHOW and HIDE let persons hide from each other selectively.

Associations work on symbolic names too. Currently used are the symbolic names "_new", "_any", and "_visible", which mean that a command creates a new object, is to be executed by all, or by all visible objects.

more...